1// $f(n) = O(g(n))$ or $f(n) = \Omega(g(n))$ or $f(n) = \Theta(g(n))$

We need to take the limit of $\frac{f(n)}{g(n)}$ as $n \to \infty$

if lim is 0 $\Rightarrow$ $f(n) = O(g(n))$,
if lim is ∞ $\Rightarrow$ $f(n) = \Omega(g(n))$,
if lim is positive constant $\Rightarrow$ $f(n) = \Theta(g(n))$

a) $f(n) = n^2+7n$, $g(n) = n^3+7$

$\lim_{n\to\infty} f(n)/g(n) = \lim_{n\to\infty} \frac{n^2+7n}{n^3+7}$

$= \lim_{n\to\infty} \frac{2n+7}{3n^2} = \lim_{n\to\infty} \frac{2}{6} = 0$

So $f(n) = O(g(n))$.

b) $f(n) = 12n + \log_2 n^2$, $g(n) = n^2+6n$

$12n + \log_2(n^2) = 12n + 2\log_2 n$,

$\lim_{n\to\infty} \frac{12n+2\log_2 n}{n^2+6n} = \lim_{n\to\infty} \frac{12 + 2\times\frac{1}{n\ln 2}}{2n+6}$

$= \lim_{n\to\infty} \frac{12 + \frac{2}{n\ln 2}}{2n+6} = \frac{12+0}{\infty} = 0$

So $f(n) = O(g(n))$.

c) $f(n) = n \log_2 3n$, $g(n) = n + \log_2(8n^3)$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n\log_2 3n}{n+\log_2(8n^3)}$$

$$= \lim_{n\to\infty} \frac{1\log_2 3n + 3n\frac{1}{3n\ln 2}}{1+\frac{3}{8n^3\ln 2}} = \frac{\log_2 3n + \frac{1}{\ln 2}}{1+\frac{3}{n\ln 2}} = \infty \text{ when } n\to\infty$$

So $f(n) = \Omega\, g(n)$

---

d) $f(n) = n^n + 5n$, $g(n) = 3.2^n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^n+5n}{3.2^n}$$

$$= \lim_{n\to\infty} e^{\log\left(\frac{2^{-n}(5n+n^n)}{3}\right)} = e^{\lim_{n\to\infty} \log(5\times 2^{-n} n + 2^{-n} n^n) - \log(3))}$$

$$= e^{\infty} = \infty \qquad \text{so} \quad f(n) = \Omega\, g(n).$$

e) $f(n) = \sqrt[3]{2n}$ $g(n) = \sqrt{3n}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\sqrt[3]{2n}}{\sqrt{3n}} = \lim_{n\to\infty} \frac{(2n)^{n/3}}{(3n)^{n/2}}$$

the denominator will grow much faster than the numerator as $n \to \infty$ so the limit is 0.

$f(n) = O(g(n))$

2/

a)

The worst-case time complexity of methodA is $O(n)$ (n is the length of the array.) Because the loop iterates over array and does a constant time operation for each element. I assumed string lengths are short and close to each other like: "CSE222", "HW2"...

b) The loop iterates over myArray which has constant length 3. It calls methodA 3 times with constant myArray.length. $O(3) \times O(1)$ ×3 simplifies to $O(1)$.

c) The code does a constant time operation (print the array[i]) n times, Its worst-case time complexity is therefore $O(n)$. int i=0; is a constant time operation (initializes it once) so I excluded it.

d) The method iterates over entire array until it finds a number greater than or equal to 4. In worst-case the method iterates over entire array. If the length of the array is n, time complexity is O(n).

3// Each method performs a constant time operation (printing) myArray.length times. So the time complexity of each method is O(n). But withoutloop method have many repeated unnecessary code which can cause typing errors and it won't be able to determine how many times it should increment the index because it doesn't use a loop. Therefore withloop method is more advantageous.

4// It is not possible to check whether it has a specific integer because the code has to iterate the array until it finds that integer.

5//

```
minA = A[0]
minB = B[0]
maxA = A[0]
maxB = A[0]

for i from 1 to n-1 do
    if A[i] < minA then
        minA = A[i]
    if A[i] > maxA then
        maxA = A[i]

for j from 1 to m-1 do
    if B[j] < minB then
```

5//

```
minA = A[0]
maxA = A[0]
minB = B[0]
maxB = B[0]

for i from 1 to n-1 do
    if A[i] < minA
        minA = A[i]
    if A[i] > maxA
        maxA = A[i]

for j from 1 to m-1 do
    if B[j] < minB
        minB = B[j]
    if B[j] > maxB
        maxB = B[j]

ans = minA * minB

if minA * maxB < ans
    ans = minA * maxB

if maxA * minB < ans
    ans = maxA * minB

if maxA * maxB < ans
    ans = maxA * maxB

return ans
```

// This algorithm has time complexity of O(n+m) which is linear. I calculated the minimum and maximum values of each array to find the minimum product.